10-30-00

A

CONLEY, ROSE & TAYON

A PROFESSIONAL CORPORATION
THE CHASE BUILDING
700 LAVACA, SUITE 800
AUSTIN, TEXAS 78701-3102
(512) 476-1400
FACSIMILE (512) 703-1250
www.intprop.com

INTELLECTUAL PROPERTY LAW
INCLUDING
PATENTS, TRADEMARKS,
COPYRIGHTS AND
UNFAIR COMPETITION

HOUSTON OFFICE
CHASE TOWER
600 TRAVIS, SUITE 1800
HOUSTON, TEXAS 77002-29
(713) 238-8000
FACSIMILE (713) 238-8008

ERIC B. MEYERTONS
(512) 703-1254
emeyertons@intprop.com

FILE: 5053-31201

October 27, 2000

"EXPRESS MAIL" MAILING LABEL

NUMBER: EL675026056US
DATE OF DEPOSIT: October 27, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to The Assistant Commissioner for Patents, Washington, D.C. 20231.

Derrick Brown

ATTN: PATENT APPLICATIONS
Assistant Commissioner for Patents
Washington, D.C. 20231

Re: U.S. Patent Application Entitled **"PROCESSING BUSINESS TRANSACTIONS USING DYNAMIC DATABASE PACKAGESET SWITCHING" (CAMS II)** -- Doughty, et al. (Atty Dkt. No. 5053-31201)

Sir:

Transmitted herewith for filing is a disclosure including a title page, a 28 page specification, 6 pages of claims (Claims 1-13), and a one page abstract. In addition, we have also included Figures 1-15 on 16 sheets. The disclosure and drawings constitute the application of Steven G. Doughty and Charles P. Bobbitt for the above-entitled invention.

Please note that this application is filed without an inventor Declaration and filing fees. Applicant requests the Patent and Trademark Office to accept this application and accord a serial number and filing date as of the date this application is deposited with the U.S. Postal Service for Express Mail. Further, the Applicant requests that the NOTICE OF MISSING PARTS-FILING DATE GRANTED be sent to the undersigned Applicant representative.

Please address all correspondence in connection with this application to:

ERIC B. MEYERTONS
CONLEY, ROSE & TAYON, P.C.
P.O. BOX 398
AUSTIN, TEXAS 78767-0398
PH: (512) 476-1400

Please date stamp and return the enclosed postcard to evidence receipt of this application.

Respectfully submitted,

Mark R. DeLuca
Reg. No. 44,649
Agent for Applicant

EBM:gm
Enclosures

PATENT
5053-31201

"EXPRESS MAIL" MAILING LABEL
NUMBER EL675026056US

DATE OF DEPOSIT: OCTOBER 27, 2000

I HEREBY CERTIFY THAT THIS PAPER OR FEE IS BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE "EXPRESS MAIL POST OFFICE TO ADDRESSEE" SERVICE UNDER 37 C.F.R. § 1.10 ON THE DATE INDICATED ABOVE AND IS ADDRESSED TO BOX PROVISIONAL PATENT APPLICATION, ASSISTANT COMMISSIONER FOR PATENTS, WASHINGTON, D.C. 20231

Derrick Brown

PROCESSING BUSINESS TRANSACTIONS USING DYNAMIC DATABASE PACKAGESET SWITCHING

By:

Steven G. Doughty
Citizenship: USA
2332 Brennan Drive
Plano, Texas 75075-6618

and

Charles P. Bobbitt
Citizenship: USA
6606 Mapleshade Lane
Dallas, Texas 78252

Atty. Dkt. No.: 5053-31201

Eric B. Meyertons
Conley, Rose & Tayon, P.C.
P.O. Box 398
Austin, Texas 78767-0398
Ph: (512) 476-1400

PRIORITY CLAIM

This application claims the benefit of U.S. Provisional Application No. 60/162,708 entitled "Processing Business Transactions Using Dynamic Database Packageset Switching," filed October 29, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to computer software programs and databases to be used in Financial Service Organizations. More particularly, the present invention relates to the dynamic selection of a database identifier, associated with a database, based on application program requirements in a Financial Service Organization (FSO) business transaction processing system.

2. Description of the Related Art

FSOs such as banks, credit unions, etc., use computer systems running software programs to process customer account transactions. The computer systems may include databases for storing data such as the master files of customer account information, transactional data such as customer credit card purchase transactions, processing data such as the processing parameters used in processing transactions, and history data such as log files of daily activities for later batch processing.

Databases may be used in FSO business transaction processing systems to store, manage and retrieve data for a variety of applications. In many instances, the databases may be extremely large. The contents of the database may often occupy memory space measured in hundred's of gigabytes. When application programs need to access

enormous amounts of data on a transactional basis, whether in batch mode or real-time mode, the FSO business transaction processing systems often utilize commercially available database products. One embodiment of a commercially available database product is the DB2 database from International Business Machines (IBM).

Some nomenclature is introduced here to aid in the understanding of terminology used. A database packageset contains information needed by the FSO database software to locate and access data in the most efficient way for a particular application program in the FSO business transaction processing system. Database packageset switching occurs when the database packageset values associated with one application program are changed to the database packageset values associated with another application program in the same business transaction processing system. Dynamic database packageset switching occurs when the database packageset values associated with one application program are changed in real-time to the database packageset values associated with another application program in the same business transaction processing system. FSO database environmental variables and database registry values may contain information to control a specific database environment within the FSO business transaction processing system. The user may change environmental variables and registry values to change the FSO database environment.

An example of an FSO that may use such a business transaction processing system is a credit card institution. A credit card institution may issue credit cards to customers of the FSO. The credit card institution may also issue credit cards on behalf of client businesses such as department stores. The credit card institution may also acquire and process credit card transactions from customers and client businesses such as department stores. For example, a credit card institution may issue its own credit card. The credit card institution may also have a client department store. The credit card institution may issue a credit card under the department store's name, and may collect and process all credit card transactions for the department store, charging a fee for each

transaction processed. Some of the credit card transactions collected by the credit card institution may be for credit cards not issued by the credit card institution. These transactions may be forwarded to the FSO that issued the card. In turn, other FSOs may forward credit card transactions to the credit card institution. Transactions for credit cards issued by the credit card institution may be processed by the credit card institution.

The FSO business transaction processing system may include a data dictionary. A data dictionary may be defined as a collection of descriptions of the data items or elements in the database. For example, the FSO business transaction processing system data dictionary may describe the data elements involved in credit card processing. The data dictionary may describe each of the data elements in the database for credit card processing. Groups of data such as master files and transactions may be comprised of data elements defined in the data dictionary. Examples of data elements in the FSO data dictionary are customer name, credit card type, and card issuer.

The FSO business transaction processing system may include processing parameters used in processing transactions. Processing parameters may be used to apply business logic to the data elements in the transaction during processing. An example of a transaction in the FSO system is a credit card transaction. An example of a processing parameter is a transaction price that may be charged to a client of a credit card institution for processing a credit card transaction. Another example of a processing parameter may be a database identifier, which may point to the location of the data in an FSO database.

The FSO transaction processing application software program may use one or more processing parameters while processing a transaction. A processing parameter may have different values for different transactions. The application software program may examine the values of one or more data elements in the transaction data or database master files to determine the value of a processing parameter for the transaction.

A combination of data elements used to determine the value of a processing parameter may be referred to as a key definition for the processing parameter. The combination of data element values constructed from the key definition may be referred to as a key value. For example, a software program for processing credit card transactions for a credit card institution may use the credit card issuer and card type to determine the database identifier.

Key definitions and key value construction are hardcoded in the source code for the FSO system software programs. Modifying the key definitions and the construction of key values from the key definitions involves modifying the source code for all software programs that use the key definitions, recompiling and relinking the programs, reinstalling the software programs, and possibly modifying the data dictionary and database structure used by the software programs. If more than one FSO use the software programs, customization of key definitions and key value construction for one of the FSOs requires creating and maintaining a customized copy of the source code for the programs.

In some FSO systems, processing parameters, such as the database identifier, and the key values used to identify them may be hardcoded in the source code for the FSO system software programs. Modifying the processing parameters and key values for these systems may involve modifying the source code for all software programs that use the processing parameter, recompiling and relinking the programs, and reinstalling the software programs. In other FSO systems, the processing parameters and key values may be stored in the FSO system database. In these systems, the processing parameters and key values in the database and the key definitions and key value construction in the source code must be synchronized. Thus, the processing parameters and key values in the database may be considered hardcoded as well, as they cannot be modified without also modifying the source code and rebuilding the programs as described above.

During the initial design and development phase of a large scale database application, such as the DB2, it may be necessary or desirable to have different database environments for Development, Quality, Testing, Production, etc. In addition, the Development environment may also be different, based on the operating system of the computer. It may be necessary or desirable to have one or more databases within one environment. For example, it may be desirable to have a separate database for each development engineer within one DB2 Development environment.

Database limitations may prohibit the use of multiple environments within the same database sub-system. The number of multiple databases within one environment within the same database sub-system may also be limited. One embodiment of a database, which may have some of these limitations is the DB2 database from International Business Machines (IBM). These limitations may often substantially reduce the flexibility of the FSO transaction processing application software program to adapt to changing business requirements. For example, an FSO business transaction processing system may need to expand and respond quickly to integrate a newly acquired FSO or process transactions from a new FSO location. A conventional method to implement these changes often involves changing the source code of the application program. This method is programming intensive, time consuming and costly.

The conventional method for the FSO transaction processing application software program developer to work around database environment restriction may be to manually modify the application program source code, in an off-line mode, to change the environment name in the database registry values. The application source code may be written in higher level programming languages such as COBOL, SQL, C, Visual Basic, C++, Java, and others. Manual switching between the one or more database environments may often involve a shutdown and startup phase for the database sub-system. This is especially undesirable in installations, which attempt to operate continuously.

The conventional method for the FSO transaction processing application software program developer to avoid the difficulty of adding a new database name may be to add new data to an existing database name. As an alternate, the application software program developer could manually modify the application program source code, in an off-line mode, to change the database name in the DB2 registry values to the 'new' database name. The application source code may be written in higher level programming languages such as COBOL, SQL, C, Visual Basic, C++, Java, and others. Both of these methods have drawbacks.

The method of adding new data to an existing DB2 database increases the size, often reducing the system performance and increasing maintenance complexity. The method of changing a database name by using applicable commands in the application programming language i.e. by modifying the source code, significantly increases the development time and the costs. For example, adding a new company database in an FSO may affect the source code in thousands of application programs.

SUMMARY OF THE INVENTION

The present invention provides various embodiments of an improved method and system for dynamically selecting a database packageset table entry based on application program requirements. In some embodiments, data may be stored in tables in the database. In one embodiment, a key definition table, database packageset table for storing database identifier values and key values may be provided.

In one embodiment, a key definition may be configured for use in building a processing key value from data associated with an FSO customer transaction in response to a computer program executing on the FSO computer system requesting a processing parameter value to be used in the processing of the FSO customer transaction. The processing key value may be used in locating the database identifier value in a dynamic database packageset table in the FSO computer system database. The key definition may also be used during the comparison of processing key values to pre-configured key values stored in a database packageset table. If a pre-configured key value that matches the processing key value is found in the dynamic database packageset table, a database identifier value stored with the pre-configured key value in the database packageset table may be returned to the requesting computer program.

The key definition may include one or more data elements, or key elements. The key elements may be arranged in a sequence in the key definition. The key elements may include information describing the location and data format of data element values in the FSO system database. The location information in a key element may be used in locating a data element value in the database during the building of a processing key value, wherein the data element value is to be included as part of the processing key value. The data format information may be used to specify the data type, size and other data attributes during the building of a processing key value. The key definition may be

stored in the database in the FSO system. In one embodiment, the key definition may be stored in a key definition table in the database.

Processing parameter values may be configured for use during the processing of data, including FSO transactions, in the FSO computer system. Key values may be configured for use in locating the processing parameters in the FSO computer system. In one embodiment, database identifier values and user defined key values may be stored in a dynamic database packageset table in the database, with one row in the dynamic database packageset table including one or more fields for storing a key value and one or more fields for storing processing parameter values associated with the key value. In one embodiment, a key definition may be used to format the user interface for entering key element values. A key element value may be entered for each key element in the key definition, and the key element values may be combined to construct a key value.

In one embodiment of a system for processing business data in an FSO transaction processing computer system using user-configured key definitions, a key building program may be provided. The key building program may be configured to accept requests for processing parameters from FSO transaction processing programs executing on the FSO computer system. The key building program may be configured to access a key definition table in the FSO system database and to read a key definition for a processing parameter in response to receiving a request for the processing parameter.

A key building program may be configured to build a processing key value for a processing parameter from one or more data element values in the FSO system database in response to receiving a request for the processing parameter. In one embodiment, the key building program may be configured to locate data element values in the FSO system database using data element location information stored in the a key definition. In one embodiment, each of one or more key elements in a key definition may specify the location and data format of one data element in the FSO system database. The key

building program may be configured to use the data element information stored in the key elements of the key definition to locate data elements in the FSO system database and to read the data element values from the FSO system database. The key building program may be configured to write each data element value read from the FSO system database to a corresponding key element in a processing key. In one embodiment, the key element values in the processing key may be in combination referred to as the processing key value.

In one embodiment of a system for processing business data in an FSO transaction processing computer system using dynamic database packageset switching, a dynamic database packageset switching software program may be provided. In one embodiment, the program may be configured to receive processing key values from a key building program. The dynamic database packageset switching software program may be configured to locate a dynamic database packageset switching table for locating a database identifier in the FSO system database in response to receiving a data value request from the application program. In one embodiment, the dynamic database packageset switching software program may compare the processing key value to the pre-configured key values stored in the rows of the dynamic database packageset switching table.

In one embodiment, the dynamic database packageset switching program may be configured to send a message to the application program in response to the dynamic database packageset switching program not finding a match for the processing key value among the pre-configured key values in the dynamic database packageset switching table for the database identifier. The message may inform the application program that no match was found for the processing key value submitted to the dynamic database packageset switching program.

In one embodiment, the key definitions, key values, processing parameter values, and database identifier values may be constructed and stored during the configuration of the FSO system. Configuration of the FSO system may occur at the time the FSO system software programs and databases are initially installed and set up for processing FSO transactions. Configuration of the FSO system may also occur after the initial configuration performed during the installation of the FSO system. A configuration of the FSO system that occurs after the initial configuration may be called a reconfiguration of the FSO system. During reconfiguration, the key definitions, key values, processing parameter values, and database identifiers constructed during the initial configuration may be modified or deleted, and new key definitions, key values, processing parameter values, and database identifiers may be added to the FSO system.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating the architecture of one embodiment of a financial service organization business transaction processing system with an application program, a database sub-system, a data storage sub-system and a user sub-system in the network;

Figure 2 is a block diagram illustrating the architecture of one embodiment of a financial service organization business transaction processing system with an application program, a dynamic database packageset switching sub-system, a computing environment monitoring sub-system, a database sub-system, a data storage sub-system, and a user sub-system in the network;

Figure 2a is a block diagram illustrating one embodiment of a dynamic database packageset switching module, a packageset input/output processing sub-system, a dynamic database packageset switching table, a database input/output processing sub-system, a computing environment monitoring sub-system, a database sub-system, a data storage sub-system;

Figure 3 illustrates a prior art database domain architecture of a financial service organization business transaction processing system with the database domain consisting of a single database sub-system, a single database environment within the database sub-system and one or more databases within the single DB2 database environment;

Figure 4 illustrates a multi-environment, multi-database domain architecture of one embodiment of a financial service organization business transaction processing system with the database domain consisting of a single database sub-system, one or more database environments within the single database sub-system and one or more databases within each of the database environments;

Figure 5 is a prior art flowchart illustrating the need for modifying application program source code to a change in database environment or an addition of a new database according to one embodiment;

Figure 5a is a continuation of flowchart in Figure 5;

Figure 6 is a flowchart illustrating the runtime use of dynamic database packageset switching method according to one embodiment;

Figure 7a is a flowchart illustrating the configuration and runtime use of dynamic database packageset switching method according to one embodiment;

Figure 7b is a continuation of flowchart in Figure 7a;

Figure 8 is a data flow diagram illustrating a dynamic database packageset switching for an associated application program database request according to one embodiment;

Figure 9 is a data flow diagram illustrating a legacy method for an application program database request according to one embodiment;

Figure 10 illustrates one embodiment of a method for selecting data dictionary data elements as key elements available for inclusion in key definitions;

Figure 11 illustrates one embodiment of a method for selecting key elements to be available for inclusion in a particular key definition from a list of key elements available for inclusion in all key definitions;

Figure 12 illustrates one embodiment of a method for selecting key elements for inclusion in a key definition from a list of key elements available for inclusion in the key definition;

Figure 13 illustrates one embodiment of a key definition with examples of parameters that may be included in the key element definitions;

Figure 14 illustrates one embodiment of a dynamic database packageset switching table for the key definition of Figure 13 with examples of key values and processing parameter values; and

Figure 15 illustrates one embodiment of a dynamic database packageset switching table in an FSO system.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and

detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE PREFERED EMBODIMENTS

The term "computer system" as used herein generally describes the hardware and software components that in combination allow the execution of computer programs. The computer programs may be implemented in software, hardware, or a combination of software and hardware. A computer system's hardware generally includes a processor, memory media, and Input/Output (I/O) devices. As used herein, the term "processor" generally describes the logic circuitry that responds to and processes the basic instructions that operate a computer system. The term "memory medium" includes an installation medium, e.g., a CD-ROM, or floppy disks; a volatile computer system memory such as DRAM, SRAM, EDO RAM, Rambus RAM, etc.; or a non-volatile memory such as optical storage or a magnetic medium, e.g., a hard drive. The term "memory" is used synonymously with "memory medium" herein. The memory medium may comprise other types of memory or combinations thereof. In addition, the memory medium may be located in a first computer in which the programs are executed, or may be located in a second computer that connects to the first computer over a network. In the latter instance, the second computer provides the program instructions to the first computer for execution. In addition, the computer system may take various forms, including a personal computer system, mainframe computer system, workstation, network appliance, Internet appliance, personal digital assistant (PDA), television system or other device. In general, the term "computer system" can be broadly defined to encompass any device having a processor that executes instructions from a memory medium.

The memory medium preferably stores a software program or programs for a dynamic database packageset switching method in a financial service organization business transaction processing system as described herein. The software program(s) may be implemented in any of various ways, including procedure-based techniques, component-based techniques, and/or object-oriented techniques, among others. For example, the software program may be implemented using ActiveX controls, C++

objects, JavaBeans, Microsoft Foundation Classes (MFC), or other technologies or methodologies, as desired. A CPU, such as the host CPU, executing code and data from the memory medium includes a means for creating and executing the software program or programs according to the methods, flowcharts, and/or block diagrams described below.

A computer system's software generally includes at least one operating system such as MVS available from International Business Machines (IBM) Corporation, a specialized software program that manages and provides services to other software programs on the computer system. Software may also include one or more programs to perform various tasks on the computer system and various forms of data to be used by the operating system or other programs on the computer system. The data may include but are not limited to databases, text files, and graphics files. A computer system's software generally is stored in non-volatile memory or on an installation medium. A program may be copied into a volatile memory when running on the computer system. Data may be read into volatile memory as the data is required by a program.

A server program may be defined as a computer program that, when executed, provides services to other computer programs executing in the same or other computer systems. The computer system on which a server program is executing may be referred to as a server, though it may contain a number of server and client programs. In the client/server model, a server program awaits and fulfills requests from client programs in the same or other computer systems. An example of a computer program that may serve as a server is Windows NT server, available from Microsoft Corporation.

<u>Figure 1 - A block diagram illustrating a prior art financial service organization business transaction processing system</u>

In Figure 1, an embodiment of a prior art financial service organization business transaction processing system 100 may include a computer system 110, an operating system 150, a user display screen 130 connected to the computer system via a link 140, and a database 170 residing on data storage 120. Computer system 110 includes memory 160 configured to store computer programs for execution on system 100, and a central processing unit (not shown) configured to execute instructions of computer programs residing on system 100. Application program 180 may be stored in memory 160. Database access requests generated by the Application program 180 are processed by the FSO database 170. If requested FSO data is available within the FSO database 170 then it retrieves the requested FSO data from the data storage 120. If requested FSO data is not available within the FSO database 170 then it may send an error message to the Application program 180. The user working at a display screen 130 may need to change the Application program 180 source code to eliminate the FSO database 170 error.

Figure 2 - A block diagram illustrating one embodiment of a financial service organization business transaction processing system utilizing a dynamic database packageset switching method

In Figure 2, an embodiment of a financial service organization business transaction processing system 200, utilizing a dynamic database packageset switching method, may include a computer system 210, an operating system 260, a user display screen 230 connected to the computer system via a link 240, a computing environment monitoring software 250 connected to a dynamic database packageset switching software 290, and a database 280 residing on data storage 220. Computer system 210 includes memory 270 configured to store computer programs for execution on system 200, and a central processing unit (not shown) configured to execute instructions of computer programs residing on system 200. Application program 295 is used to interface to and process financial services organization business transactions requested by a user working at a user display screen 230. Application program 295 may be stored in memory 260.

A user working at display screen 230 may execute a transaction associated with Application program 295. The transaction may need data from FSO database 280. FSO database 280 access requests from the Application program 295 are processed by the dynamic database packageset switching module 290. The dynamic database packageset switching module 290 dynamically selects the packageset name associated with the application program. The packageset specified memory location of the requested FSO data and forwards that information to the FSO database 280. The requested FSO data within the FSO database 280 is then retrieved from the data storage 220.

<u>Figure 3 - A prior art database domain architecture of a financial service organization business transaction processing system</u>

Figure 3 illustrates a prior art database domain architecture of a financial service organization business transaction processing system with the database domain consisting of a single database sub-system 300, a single database environment 310 within the database sub-system 300 and one or more databases 320 within the single database environment 310. Prior art database design and set up restrictions normally limit the financial service organization business transaction processing system to use the single environment within the same DB2 sub-system.

<u>Figure 4 - A multi-environment, multi-database domain architecture of one embodiment of a financial service organization business transaction processing system</u>

Figure 4 illustrates a multi-environment, multi-database domain architecture of one embodiment of a financial service organization business transaction processing system with a single database sub-system, one or more database environments within the single database sub-system and one or more databases within one of the database environments. The dynamic database packageset switching method and system is capable

of processing, at run time, an FSO database access request from the Application Program which would be independent of the current database environment and database name.

Figure 5 - A Flowchart illustrating the prior art for processing a business transaction which involves a change in database environment according to one embodiment;

Figure 5 is a flowchart illustrating the prior art for processing an FSO business transaction. If the Application Program associated with the business transaction requests data not available in the current database environment or in the current database name then the Application Program source code must be modified. Modifying an application program source code to accommodate a change in database environment or an addition of a new database, according to one embodiment, is time consuming and expensive. In one embodiment, steps 500 through 505 may be performed in a business transaction processing program in the FSO system. The user of the FSO computer may START the prior art FSO processing method by initiating a business transaction. In step 500, the user initiated, or another FSO computer initiated in another embodiment, business transaction may communicate with an application program. In step 501, in response to the user requested business transaction, the application program may generate one or more requests for data from FSO computer database. In step 502, a determination may be made if the requested data is obtainable within the definition of the current environment. If the requested data may be obtained in the current environment then program control is passed to step 503. If the requested data may not be obtained in the current environment then program control is passed to step 505. In step 505, the user of the FSO computer system may need to change the application program source code, in an off-line mode, to modify the environment registry values used in FSO database definition. In step 503, a determination may be made if the requested data is obtainable within the current database name. If the requested data may be obtained in the current database name then program control may be passed to step 510 in Figure 5a. In step 504, the user of the FSO computer

system may need to change the application program source code, in an off-line mode, to modify the database registry values used in FSO database definition.

Figure 5a - A Continuation of Flowchart illustrated in Figure 5

Figure 5a is a continuation of flowchart illustrated in Figure 5. In step 510, in one embodiment, the request for data may be passed on to a database manager in the FSO computer system to locate the requested data. In step 520, in one embodiment, the requested data is retrieved and passed on to the originating application program.

Figure 6 – A Flowchart illustrating the runtime version of a dynamic database packageset switching method according to one embodiment

Figure 6 is a flowchart illustrating the runtime version of the dynamic database packageset switching method for performing database access during the processing an FSO business transaction. If the Application program associated with the business transaction requests data which may not be available in the current database environment or in the current database name then the dynamic database packageset switching software dynamically changes the environment or the database name without requiring a source code change to access the requested data. In one embodiment, steps 600 and 603 may be performed in a business transaction processing program in the FSO system. The user of the FSO computer may START the dynamic database packageset switching method for performing database access during the processing an FSO business transaction by initiating a business transaction. In step 600, the user requested business transaction may initiate an application program. In step 601, in response to the user requested business transaction, the application program may generate one or more requests for data from the FSO computer database. In step 602, the dynamic database packageset switching method for performing database access may obtain the requested data from the FSO database without changing the application program source code. Further detail of step 602 is

shown in Figure 7b. In step 603, the dynamic database packageset switching method may return the requested data to the application program.

Figure 7a - A flowchart illustrating one embodiment of the configuration and runtime use of dynamic database packageset switching method

Figure 7a is a flowchart illustrating the configuration and runtime version of the dynamic database packageset switching method for performing database access during the processing an FSO business transaction. In one embodiment, steps 700 through 720 may be performed in a business transaction processing program in the FSO system. The user of the FSO computer may START the configuration of the dynamic database packageset switching method by defining values in a dynamic database packageset switching table. In one embodiment, the dynamic database packageset switching table may consist of data values stored in one or more rows and two columns. Column number one may comprise a database identifier and column number two may comprise a user defined key. The user defined key structure may be defined by selecting one or more data dictionary elements arranged in a specific sequence. The database identifier in column number one may comprise a pointer to the FSO computer database location associated with the value of the user defined key in column number two. The dynamic database packageset switching table configuration method is further detailed in Figures 11 through 16. In step 710, the dynamic database packageset switching software may be compiled, generated and downloaded. In step 720, the user may execute the dynamic database packageset switching software.

Figure 7b - A continuation of flowchart in Figure 7a

Figure 7b is the continuation of flowchart in Figure 7a. In step 722, the dynamic database switching software receives a request for data stored in the FSO database from the application program. In step 723, the dynamic database switching software program

may build a key in real-time, consistent with the user defined key structure defined in step 700. The key may then be used to access the dynamic database switching table to find a database identifier with a matching entry in the user defined key value. In step 724, the dynamic database switching software program may use the database identifier to request data from FSO database. On receiving the requested data from the FSO database, program control may transfer to step 725. The dynamic database switching software program may send the requested data to the application program in step 725. This completes the data request/response cycle for the application program, using the dynamic database switching software program. In step 726, the dynamic database switching software program may constantly monitor if the FSO computer is on-line. If it is on-line, program control transfers to step 722.

<u>Figure 8 – A data flow diagram illustrating dynamic database packageset switching for an associated application program database request according to one embodiment</u>

Figure 8 is a data flow diagram illustrating dynamic database packageset switching for an associated application program database request, according to one embodiment. In step 800, a user requests an FSO business transaction. In step 810, an application program associated with the business transaction processes the user request and generates a request for FSO data, which may be required to complete the business transaction, to the dynamic database packageset switching software. In step 820, the input processing portion of the dynamic database packageset switching software may gather additional transaction related information regarding the application program request for FSO data. In one embodiment, this may include the application program name, program variable, location of user and similar other. In step 840, the dynamic database packageset switching software may dynamically create a key to access the dynamic database packageset switching table. In steps 850 and 860, the dynamic database packageset switching software may find a matching user defined key with an associated database identifier. In step 870, the dynamic database packageset switching software may use the

database identifier to point to the physical data storage location in the FSO database. In steps 880 and 830, the requested data from FSO database may be returned to the dynamic database packageset switching software which may be passed on to the application program in step 830. The application program may then display requested data to user to complete the user business transaction.

Figure 9 – A data flow diagram illustrating a legacy method for an application program database request according to one embodiment

Figure 9 is a data flow diagram illustrating a legacy method for an application program database request, according to one embodiment. In step 900, a user requests an FSO business transaction. In step 910, an application program associated with the business transaction may process the user request and may generate a request for FSO data, which may be required to complete the business transaction. In step 920, the application to specific database interface program may convert the data request to a format understood by the FSO database. In one embodiment, this may include converting the application program data read request written in COBOL to an IBM DB2 database format. In step 930, the requested data from FSO database may be returned to the application to specific database interface program which may be pass on the data to the application program. The application program may then display requested data to user to complete the user business transaction.

Figure 10 - One embodiment of a method for selecting data dictionary data elements as key elements available for inclusion in key definitions

Figure 10 illustrates one embodiment of a method for a user of an FSO system to select data dictionary data elements as key elements available for inclusion in key definitions in an FSO system. In this illustration, by way of example, the letters A through Z are used to represent the data elements in the universe of data elements in the

data dictionary. In one embodiment, the database may include a table including references to all data elements in the data dictionary that are useable in keys. In another embodiment, each data element in the data dictionary may have a useable in keys parameter associated with it. In one embodiment, the useable in keys parameter may be a binary parameter, and may be set to either allow the data element to be used in key definitions or to exclude the data element from use in key definitions. In one embodiment, a particular data element in the data dictionary may be selected, and information about the selected data element including the useable in keys parameter may be presented on a computer display screen. In one embodiment, a list 122 of all data elements in the data dictionary may be presented on a computer display screen 120 to a user of the FSO system. In one embodiment, a current state 124 of the useable in keys parameter may be displayed with each data element. The state of the useable in keys parameter may be displayed in any form suitable to represent a binary parameter. Examples of forms of displaying states of binary parameters include, but are not limited to: textual binary displays such as YES/NO, Y/N, TRUE/FALSE, T/F, 1/0 and ON/OFF, and; graphic binary displays, such as check boxes and radio buttons. In this example, YES/NO is used, with YES representing a useable in keys parameter set to allow the data element to be used in key definitions, and NO representing a useable in keys parameter set to exclude the data element from use in key definitions. A user may change the state of a useable in keys parameter for a data element by changing the displayed state of the useable in keys parameter. In one embodiment, the user may select the useable in keys parameter using a cursor control device and enter the textual representation of the desired state of the useable in keys parameter using an input device. In one embodiment, the user may select data dictionary data elements as key elements available for inclusion in key definitions to monitor the FSO computing environment.

<u>Figure 11 - One embodiment of a method for selecting key elements to be available for inclusion in a particular key definition from a list of key elements available for inclusion in all key definitions</u>

Figure 11 illustrates one embodiment of a method for a user of an FSO system to select a group of key elements available for inclusion as key elements in a particular key definition in the FSO system. In this illustration, by way of example, the letters A through Z are used to represent the data elements in the universe of data elements in the data dictionary. In one embodiment, a list 132 of all key elements available for inclusion in key definitions may be presented on a computer display screen 130 to a user of the FSO system. In this example, list 132 includes key elements N through Z. Each key element in list 132 may have a useable in key parameter 134 associated with it. In one embodiment, a current state 134 of the useable in key parameter may be displayed with each key element. The state of the useable in key parameter may be displayed in any form suitable to represent a binary parameter. In this example, YES/NO is used, with YES representing a useable in key parameter set to allow the key element to be used in this key definition, and NO representing a useable in key parameter set to exclude the data element from use in this key definition. A user may change the state of a useable in key parameter for a key element by changing the displayed state of the useable in key parameter. In one embodiment, the user may select the useable in key parameter using a cursor control device and enter the textual representation of the desired state of the useable in key parameter using an input device.

<u>Figure 12 - One embodiment of a method for selecting key elements for inclusion in a key definition from a list of key elements available for inclusion in the key definition</u>

Figure 12 illustrates one embodiment of a method for a user of an FSO system to select key elements for inclusion in a key definition from a list of available key elements for the key definition in the FSO system. In this illustration, by way of example, the letters A through Z are used to represent the data elements in the universe of data elements in the data dictionary. In one embodiment, a list 142 of all key elements available for inclusion in a particular key definition may be presented on a computer

display screen 140 to a user of the FSO system. In this example, list 142 includes key elements V through Z. Computer display screen 140 may also display a "use in this key" parameter 144 and a key element sequence parameter 146 for each key element displayed. The state of the "use in this key" parameter 144 may be displayed in any form suitable to represent a binary parameter. In this example, YES/NO is used, with YES representing a key element selected to be used as a key element in this key definition, and NO representing a key element not selected to be used as a key element in this key definition. A user may change the state of the "use in this key" parameter for a key element by changing the displayed state of the "use in this key" parameter. In one embodiment, the user may select the "use in this key" parameter using a cursor control device and enter the textual representation of the desired state of the "use in this key" parameter using an input device. In this example, the "use in this key" parameter for key elements W, X and Z are set to YES to indicate that W, X and Z are to be used as key elements in the key definition. The key element sequence parameter may be used to specify the order in which the key elements will appear in the key definition. In this example, key element X is set to appear as the first key element, key element W is set to appear as the second key element, and key element Z is set to appear as the third key element.

<u>Figure 13 - One embodiment of a key definition with examples of parameters that may be included in the key element definitions</u>

Figure 13 illustrates an embodiment of a key definition that may have been defined by a user of an FSO system, with key elements displayed on a key definition screen 150. Each key element may include several parameters that define the key element. In one embodiment, the key elements may be displayed as rows on computer display screen 150, with the columns displaying key element parameters. This example shows key elements X, W, and Z. Key element column 152 displays the key element name. Key element sequence column 154 displays the order in which the key elements

may appear in the key definition. In this example, key element X is the first key element, key element W is the second key element, and key element Z is the third key element. Element name column 156 may display a data element name. In this example, key element X is Company ID, key element W is the Credit Card Type, and key element Z is ON US/NOT ON US. Field length column 158 may display a length in units for the key element. In one embodiment, the units are 8-bit bytes. In this example, key element X is 2 bytes long, key element W is 3 bytes long, and key element Z is one byte long. Data type column 160 may display a data type for the key element. In this example, key element X is of data type numeric, and key elements W and Z are of data type character.

Figure 14 - One embodiment of a dynamic database packageset switching table for the key definition of Figure 13 with examples of key values and processing parameters

Figure 14 illustrates an embodiment of a dynamic database packageset switching table 170 from a database used in an FSO system, with rows including user defined key values 174 and database identifiers 178 associated with the key values. The dynamic database packageset switching table may be used to store user defined key values and the database identifier values associated with the key values. The user defined key values and the database identifier may be entered by a user of the FSO system. In one embodiment, the dynamic database packageset switching table may include pre-defined key values and the database identifier, and the user of the FSO system may add key values and the database identifier to the dynamic database packageset switching table. In one embodiment, the user defined keys may be configured to monitor the FSO computing environment.

The dynamic database packageset switching table may be searched for a particular user defined key value to find the database identifier associated with the key value. In this example, the dynamic database packageset switching table 170 may be used to point to database locations 179 for different key values. The ability for a user of

the FSO system to configure key definitions and dynamic database packageset switching tables at configuration time allows the FSO application programs to look up database location for data in a particular transaction based upon attributes of the transaction. For example, an FSO may add a new city branch 175 to process business transactions for Location 14. The user may include one or more attributes (data elements) of the transaction as key elements in the key definition and define a database pointer 179. In this example, the FSO may be able to add new locations, such as adding Location 14, and process business transactions by merely configuring the dynamic database packageset switching table 170 and without changing any application program source code.

In one embodiment, each row 172 in table 170 holds one key value and its associated database identifier value. In one embodiment, each key value is unique among the key values in the dynamic database packageset switching table. Each key definition is associated with one database identifier value in the dynamic database packageset switching table. A key value may be constructed from the key element values stored in the one or more key elements defined in the key definition for this dynamic database packageset switching table. In this example, the key values are constructed from key elements X (175), W (176), and Z (177), as defined in key description 150 illustrated in Figure 14. In Figure 15, row 1's key value is (12, VIS, Y). The database identifier value corresponding to the key value of (12, VIS, Y) is NN1234567890123456. Searching the dynamic database packageset switching table 170 for the key value of (12, VIS, Y) will return the database identifier value of NN1234567890123456.

Figure 15 - One embodiment of a dynamic database packageset switching table in an FSO system

Figure 15 illustrates one embodiment of the dynamic database packageset switching table 210 in an FSO system database. Dynamic database packageset switching table 210 may include cells 216 for storing user-defined key values 217 in one column

212 and cells 218 for storing user-defined database identifier values 218 in a second column 214. One row in each table may include one cell 216 for storing a user-defined key value 217 and one cell 218 for storing the user-defined database identifier value 219 associated with the key value. In this example, in Dynamic database packageset switching table, user defined key values (1, 2, … , n) correlate to database identifiers (1, 2, …, n).

In one embodiment, cells 216 for storing key values 217 may be of a pre-configured fixed size that is identical for all dynamic database packageset switching tables 210. In this embodiment, the fixed size of cells 216 may be pre-configured large enough to store key values 217 of the maximum size anticipated by the user of the system. In one embodiment, if the pre-configured size of cells 216 is not large enough, the dynamic database packageset switching tables may be re-configured with a larger fixed size for cells 216.

Various embodiments further include receiving or storing instructions and/or data implemented in accordance with the foregoing description upon a carrier medium. Suitable carrier media include memory media or storage media such as magnetic or optical media, e.g., disk or CD-ROM, as well as signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as networks and/or a wireless link.

Although the system and method of the present invention have been described in connection with several embodiments, the invention is not intended to be limited to the specific forms set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents as can be reasonably included within the spirit and scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method performed in a Financial Service Organization (FSO) computer system, the method comprising:

building a first key value from one or more data element values stored in a first memory in the FSO computer system;

comparing the first key value to one or more key values stored in a second memory, wherein the second memory comprises one or more database identifier values each corresponding to a respective key value of the one or more key values;

writing into a third memory a first database identifier value of the one or more database identifier values stored in the second memory in response to finding a match between the first key value and one of the one or more key values stored in the second memory; and

accessing a first database in response to writing the first database identifier value into the third memory;

wherein the one or more key values and the one or more database identifier values stored in the second memory are entered by a user of the FSO computer system during a configuration of the FSO computer system.

2. The method of claim 1, wherein the FSO computer system comprises a plurality of databases, wherein the plurality of databases includes the first database, wherein each of the plurality of databases corresponds to a respective database identifier value, wherein one of the plurality of databases is an active database, wherein an active database identifier value corresponding to the active database is stored in a fourth memory, wherein the accessing the first database in response to writing the first database identifier value into the third memory comprises:

comparing the first database identifier value in the third memory to the active database identifier value in the fourth memory; and

setting the active database to the first database in response to the first database identifier value in the third memory not matching the active database identifier value in the fourth memory.

3. The method of claim 2, wherein setting the active database to the first database comprises setting the active database identifier value stored in the fourth memory to the first database identifier value from the third memory.

4. The method of claim 1, wherein the FSO computer system comprises a key definition comprising one or more data elements, wherein the first key value comprises one or more key fields, wherein the building the first key value from one or more data element values in the first memory in the FSO computer system comprises:

reading a first data element value from the first memory, wherein a location of the first data element value in the first memory is defined by a first data element from the key definition; and

storing the first data element value in a first key field in the first key value in response to reading the first data element from the first memory.

5. A carrier medium comprising program instructions, wherein the program instructions are executable by a FSO computer system to implement:

building a first key value from one or more data element values stored in a first memory in the FSO computer system;

comparing the first key value to one or more key values stored in a second memory, wherein the second memory comprises one or more database identifier values each corresponding to a respective key value of the one or more key values;

writing into a third memory a first database identifier value of the one or more database identifier values stored in the second memory in response to

finding a match between the first key value and one of the one or more key values stored in the second memory; and

accessing a first database in response to writing the first database identifier value into the third memory;

wherein the one or more key values and the one or more database identifier values stored in the second memory are entered by a user of the FSO computer system during a configuration of the FSO computer system.

6. The carrier medium of claim 5, wherein the FSO computer system comprises a plurality of databases, wherein the plurality of databases includes the first database, wherein each of the plurality of databases corresponds to a respective database identifier value, wherein one of the plurality of databases is an active database, wherein an active database identifier value corresponding to the active database is stored in a fourth memory, wherein the accessing the first database in response to writing the first database identifier value into the third memory comprises:

comparing the first database identifier value in the third memory to the active database identifier value in the fourth memory; and

setting the active database to the first database in response to the first database identifier value in the third memory not matching the active database identifier value in the fourth memory.

7. The carrier medium of claim 5, wherein setting the active database to the first database comprises setting the active database identifier value stored in the fourth memory to the first database identifier value from the third memory.

8. The carrier medium of claim 5, wherein the FSO computer system comprises a key definition comprising one or more data elements, wherein the first key value comprises one or more key fields, wherein the building the first key value from

one or more data element values in the first memory in the FSO computer system comprises:

reading a first data element value from the first memory, wherein a location of the first data element value in the first memory is defined by a first data element from the key definition; and

storing the first data element value in a first key field in the first key value in response to reading the first data element from the first memory.

9. The carrier medium of claim 5, wherein the carrier medium is a memory medium.

10. A system for processing FSO transactions, the system comprising:

a computer program;

an FSO computer system;

wherein the computer program is executable on the FSO computer system to execute:

building a first key value from one or more data element values stored in a first memory in the FSO computer system;

comparing the first key value to one or more key values stored in a second memory, wherein the second memory comprises one or more database identifier values each corresponding to a respective key value of the one or more key values;

writing into a third memory a first database identifier value of the one or more database identifier values stored in the second memory in response to finding a match between the first key value and one of the one or more key values stored in the second memory; and

accessing a first database in response to writing the first database identifier value into the third memory;

wherein the one or more key values and the one or more database identifier values stored in the second memory are entered by a user of the FSO computer system during a configuration of the FSO computer system.

11. The method of claim 10, wherein the FSO computer system comprises a plurality of databases, wherein the plurality of databases includes the first database, wherein each of the plurality of databases corresponds to a respective database identifier value, wherein one of the plurality of databases is an active database, wherein an active database identifier value corresponding to the active database is stored in a fourth memory, wherein the accessing the first database in response to writing the first database identifier value into the third memory comprises:

comparing the first database identifier value in the third memory to the active database identifier value in the fourth memory; and

setting the active database to the first database in response to the first database identifier value in the third memory not matching the active database identifier value in the fourth memory.

12. The method of claim 11, wherein setting the active database to the first database comprises setting the active database identifier value stored in the fourth memory to the first database identifier value from the third memory.

13. The method of claim 10, wherein the FSO computer system comprises a key definition comprising one or more data elements, wherein the first key value comprises one or more key fields, wherein the building the first key value from one or more data element values in the first memory in the FSO computer system comprises:

reading a first data element value from the first memory, wherein a location of the first data element value in the first memory is defined by a first data element from the key definition; and

storing the first data element value in a first key field in the first key value in response to reading the first data element from the first memory.

ABSTRACT OF THE DISCLOSURE

A system and method for dynamic selection of a database identifier, associated with a database, based on application program requirements in a Financial Service Organization (FSO) business transaction processing system. Developers may build application programs for FSO business transactions with increased flexibility to changing business requirements and reduced development time. The dynamic database packageset software, which may comprise of a dynamic database packageset table, may provide functionality to isolate the FSO application program source code from changes to the FSO database. The dynamic database packageset switching table may include user defined keys and their associated database identifier values. Dynamic database packageset switching software, used in processing FSO database requests from business transactions, may provide functionality to build a key, in real-time with a pre-defined structure, based on application program parameter values, use the key to access the dynamic database packageset switching table to locate an associated database identifier and access the FSO database using the database identifier to retrieve requested FSO data.

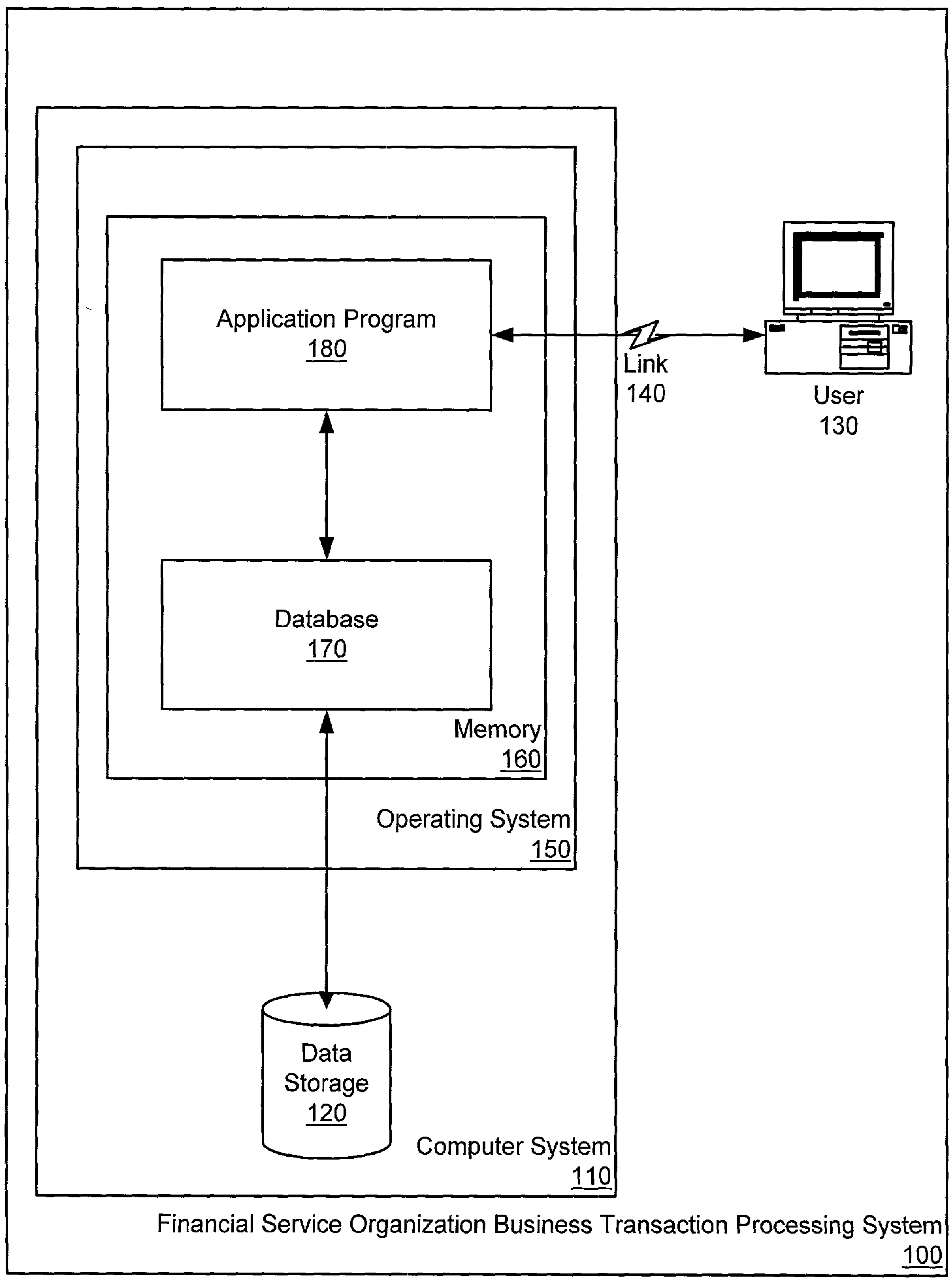


Figure 1

Prior Art

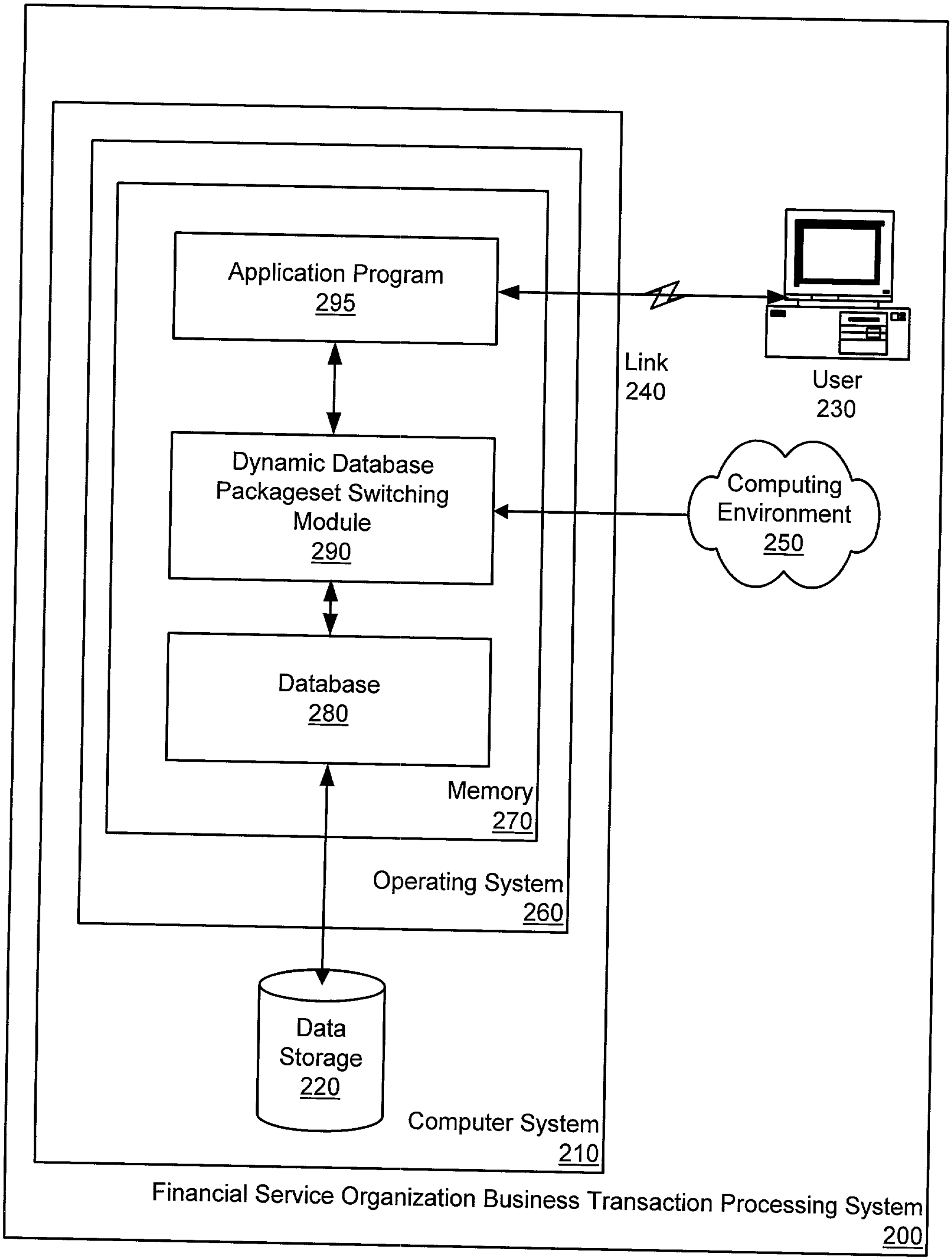
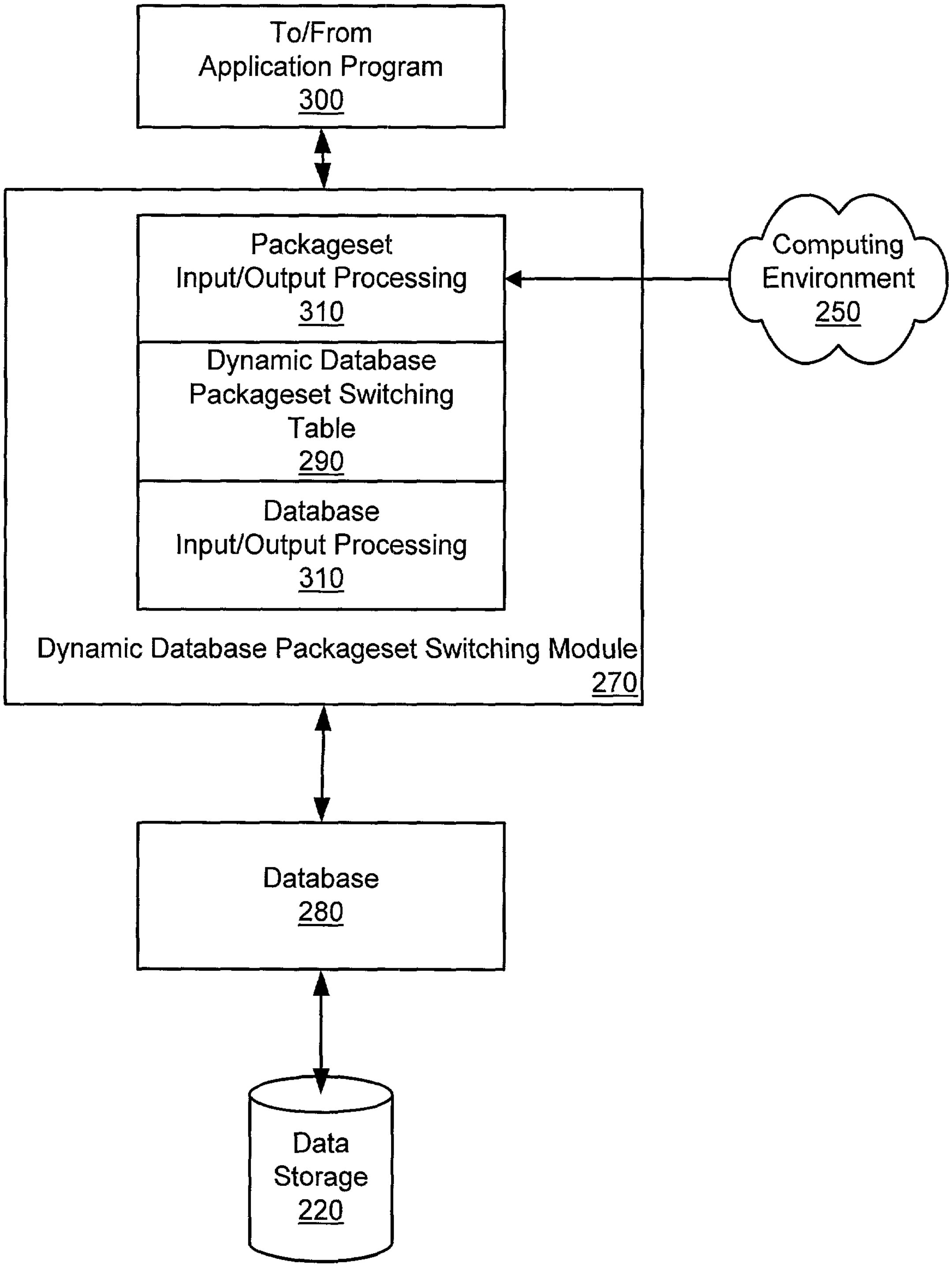

Application Program
295
Dynamic Database Packageset Switching Module
290
Database
280
Memory
270
Operating System
260
Data Storage
220
Computer System
210
Link
240
User
230
Computing Environment
250
Financial Service Organization Business Transaction Processing System
200

Figure 2



To/From
Application Program
300
Packageset
Input/Output Processing
310
Dynamic Database
Packageset Switching
Table
290
Database
Input/Output Processing
310
Dynamic Database Packageset Switching Module
270
Computing
Environment
250
Database
280
Data
Storage
220

Figure 2a

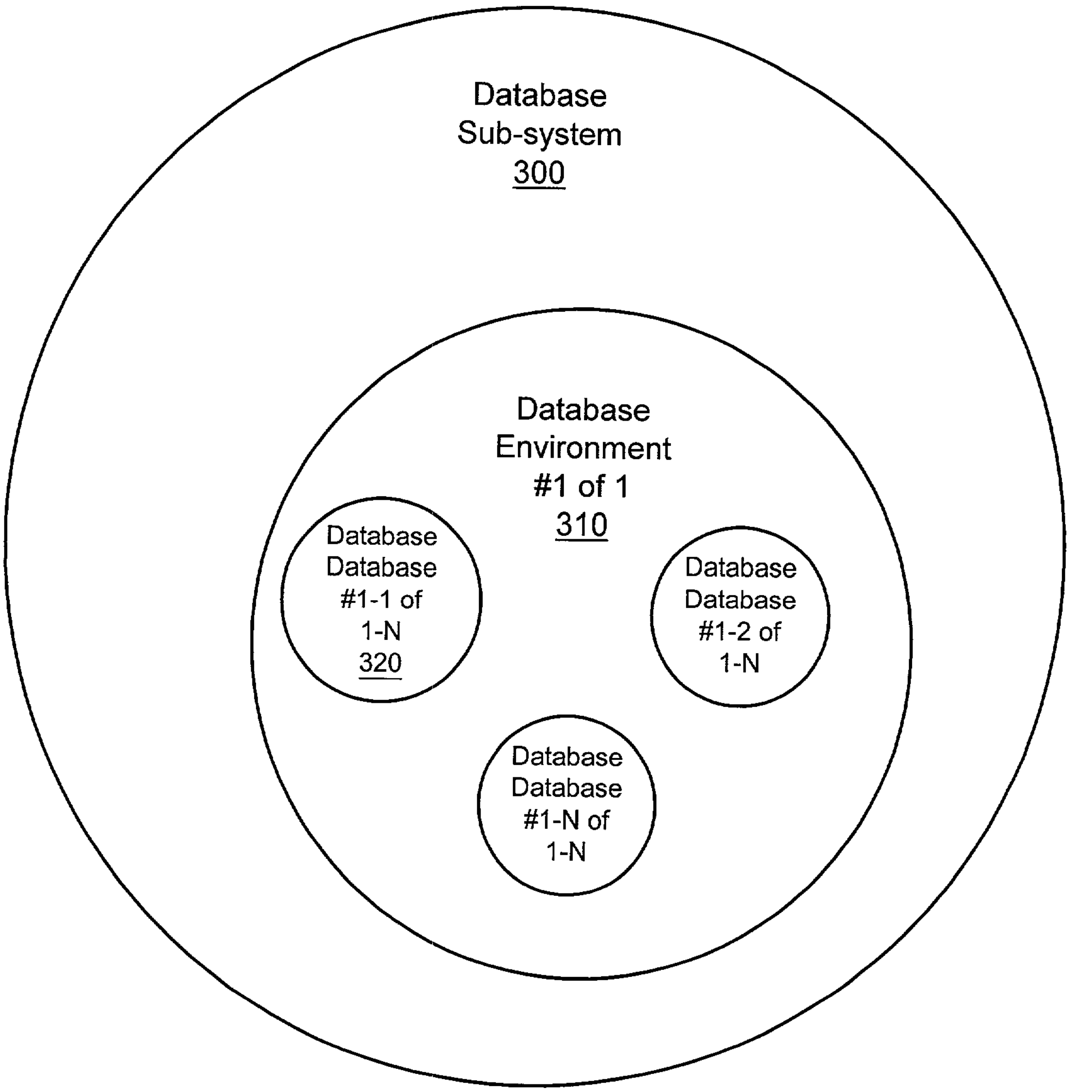


Figure 3

Prior Art

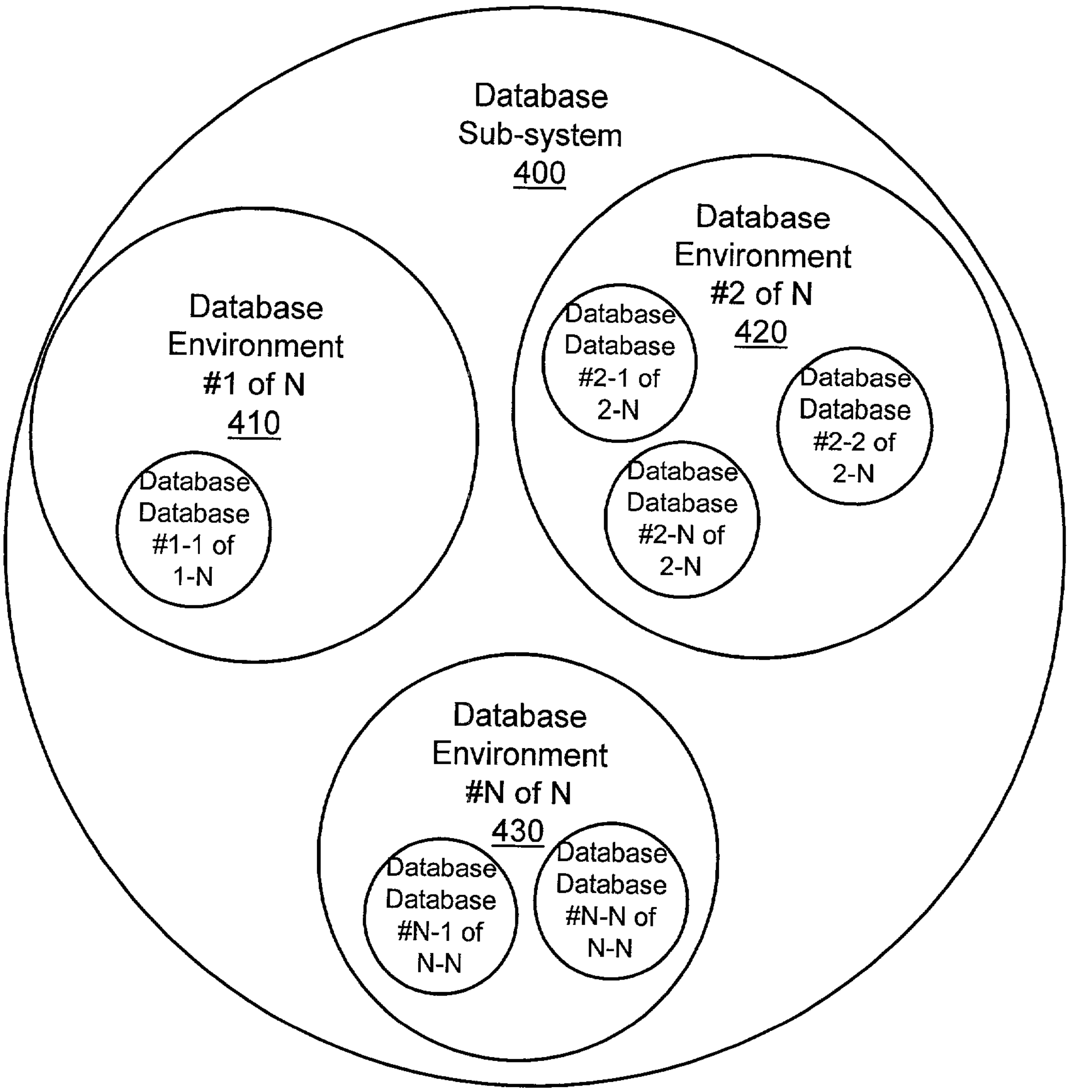

Database
Sub-system
400
Database
Environment
#1 of N
410
Database
Database
#1-1 of
1-N
Database
Environment
#2 of N
420
Database
Database
#2-1 of
2-N
Database
Database
#2-2 of
2-N
Database
Database
#2-N of
2-N
Database
Environment
#N of N
430
Database
Database
#N-1 of
N-N
Database
Database
#N-N of
N-N

Figure 4

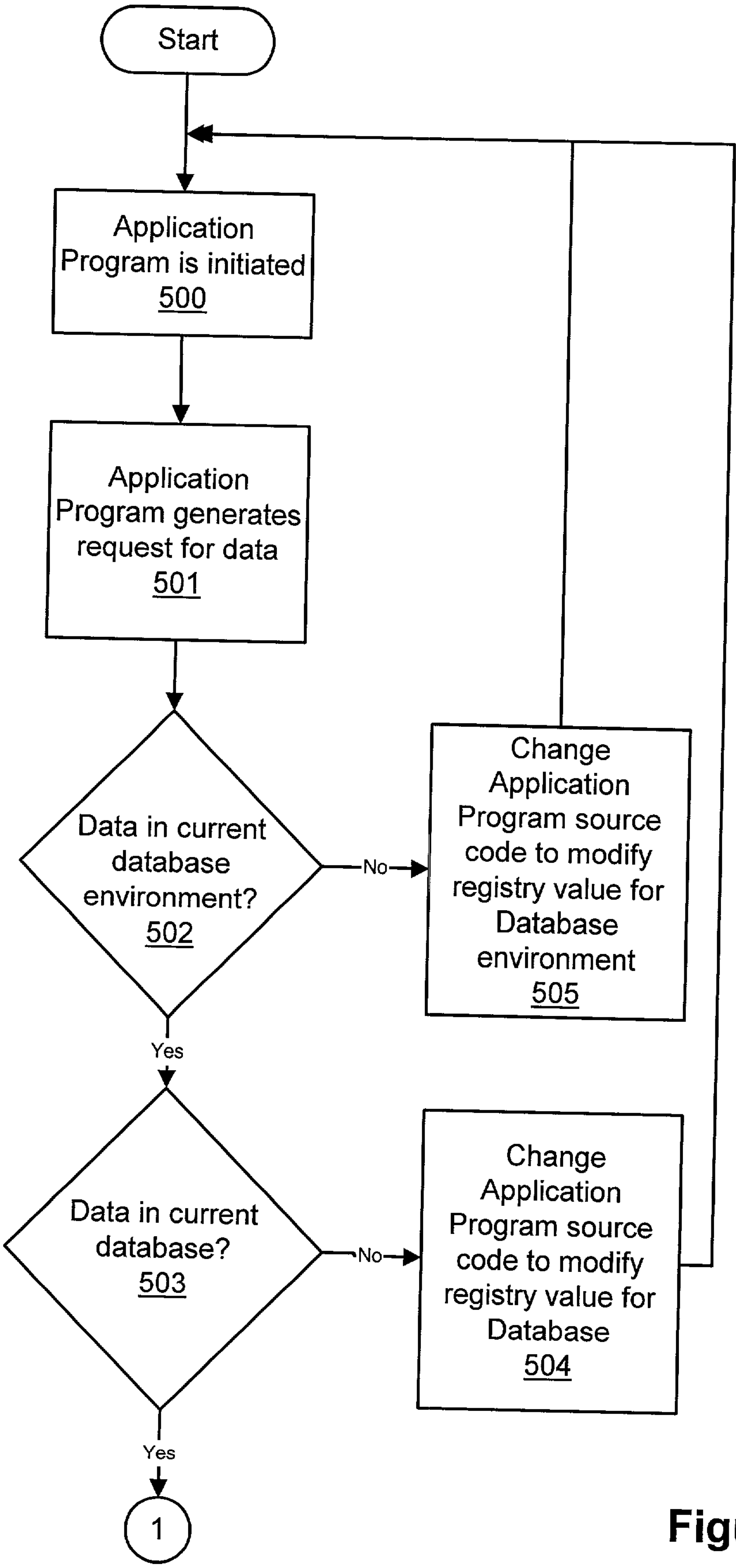


Figure 5

Prior Art

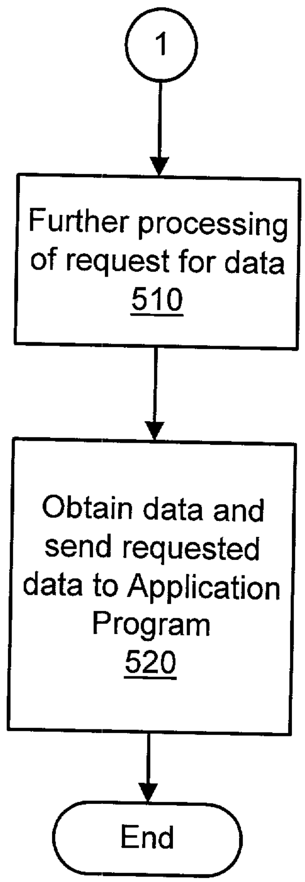

1
Further processing of request for data 510
Obtain data and send requested data to Application Program 520
End

Figure 5a

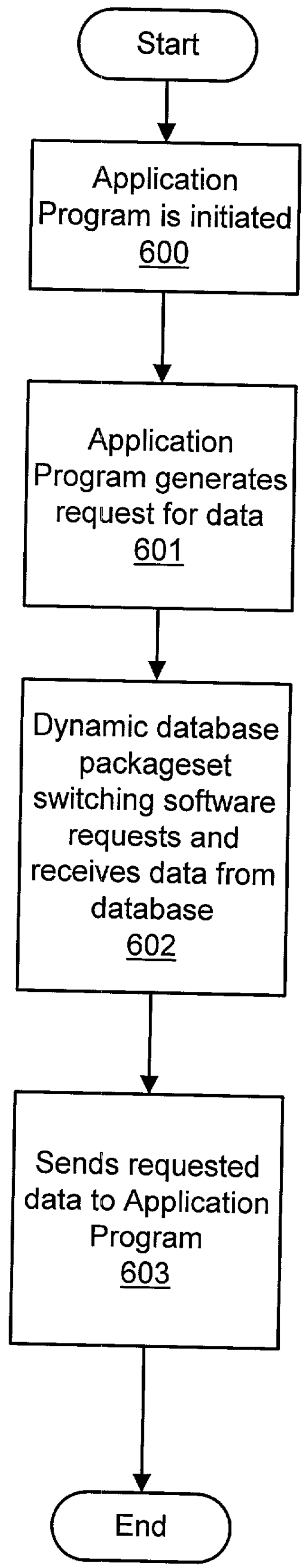

Start
Application Program is initiated 600
Application Program generates request for data 601
Dynamic database packageset switching software requests and receives data from database 602
Sends requested data to Application Program 603
End

Figure 6

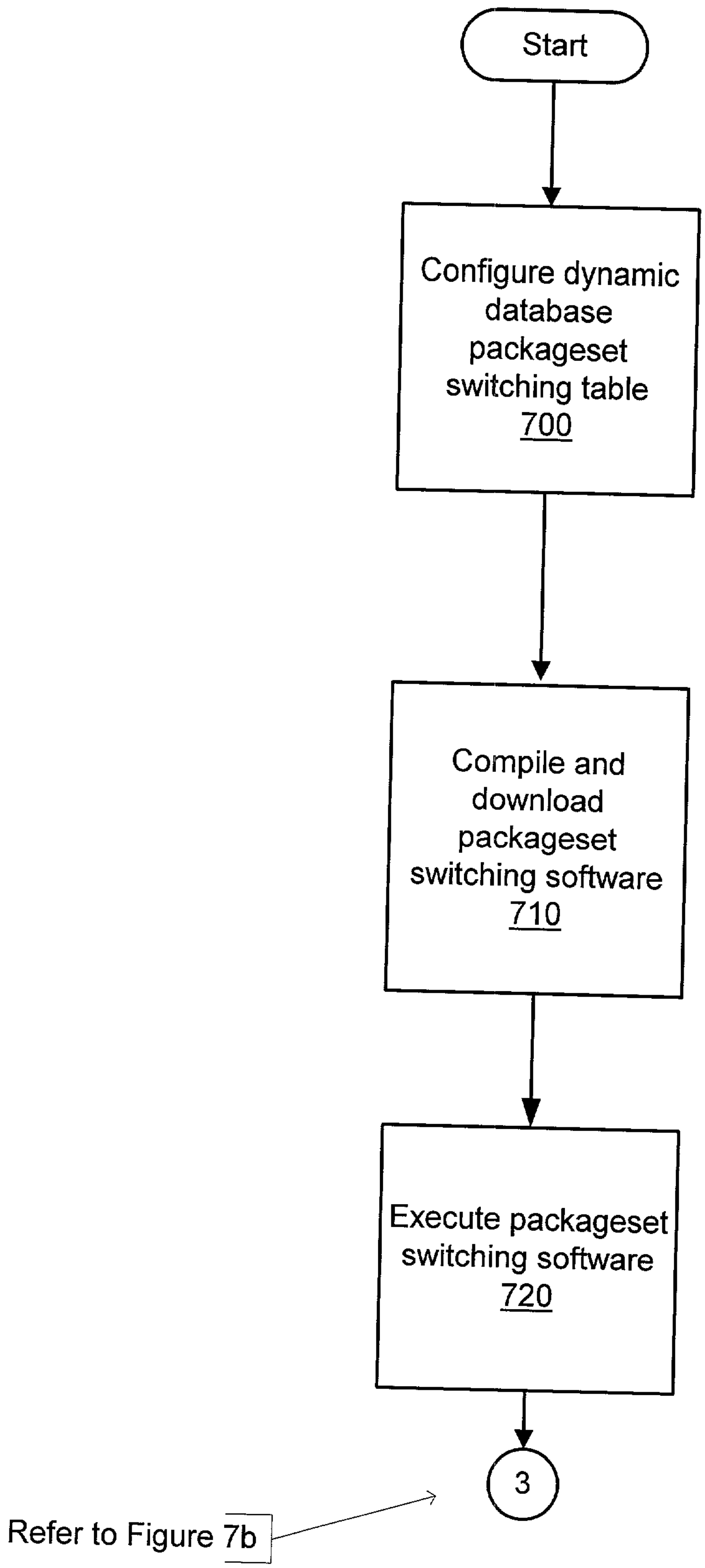

Start
Configure dynamic database packageset switching table 700
Compile and download packageset switching software 710
Execute packageset switching software 720
3
Refer to Figure 7b

Figure 7a

Detail of 602
Figure 6

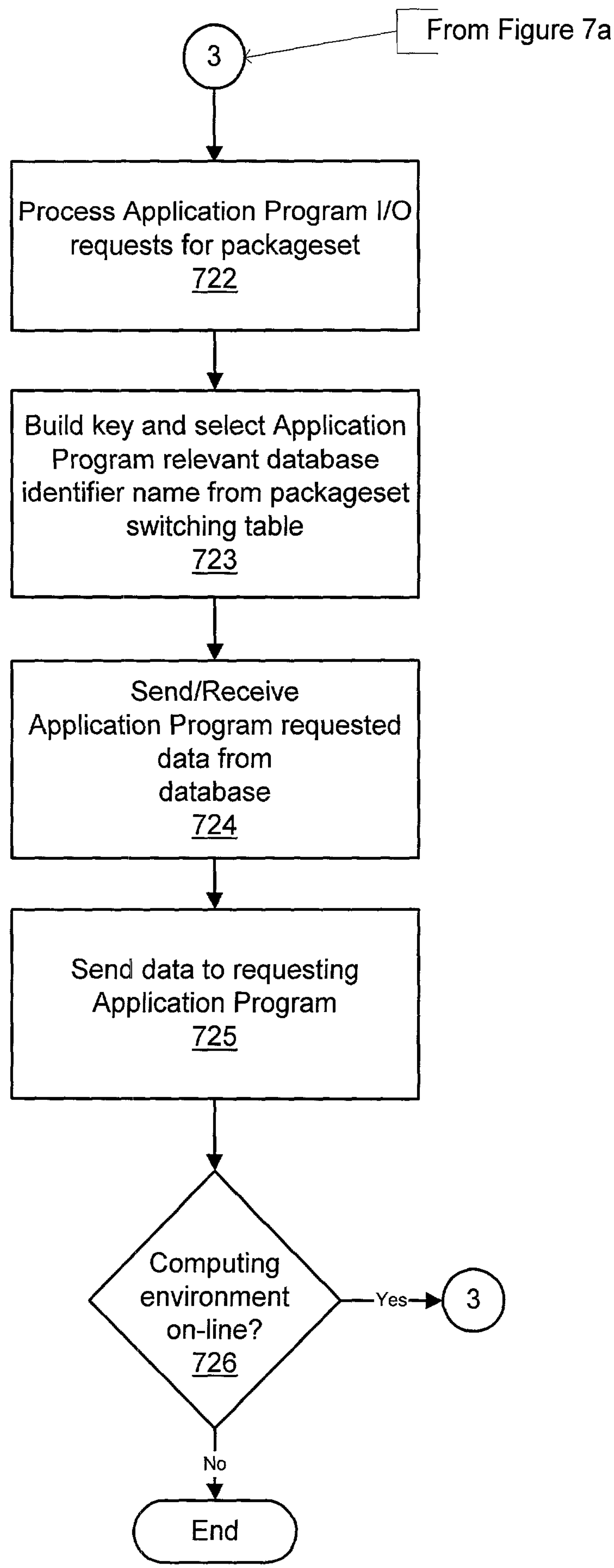


Figure 7b

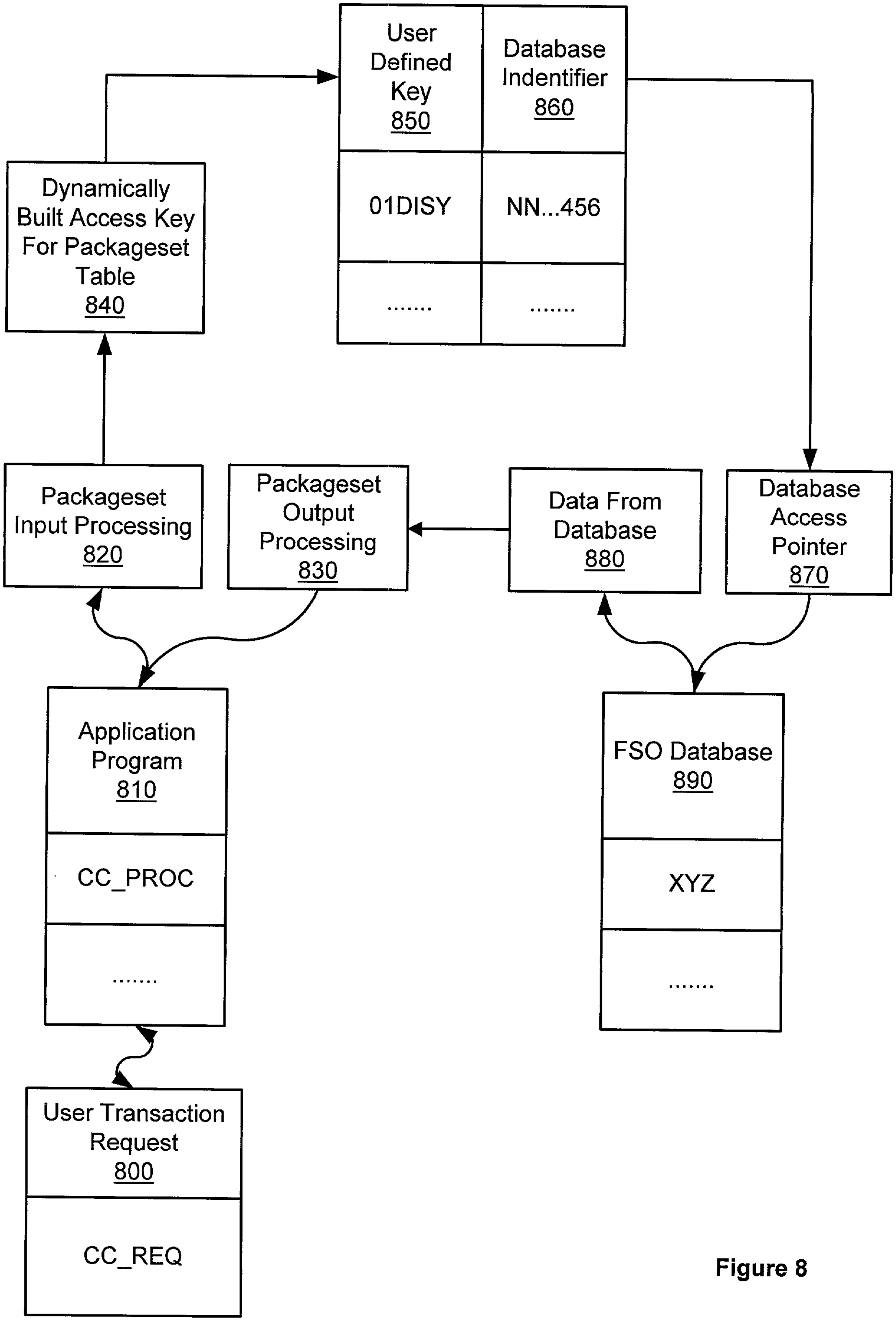
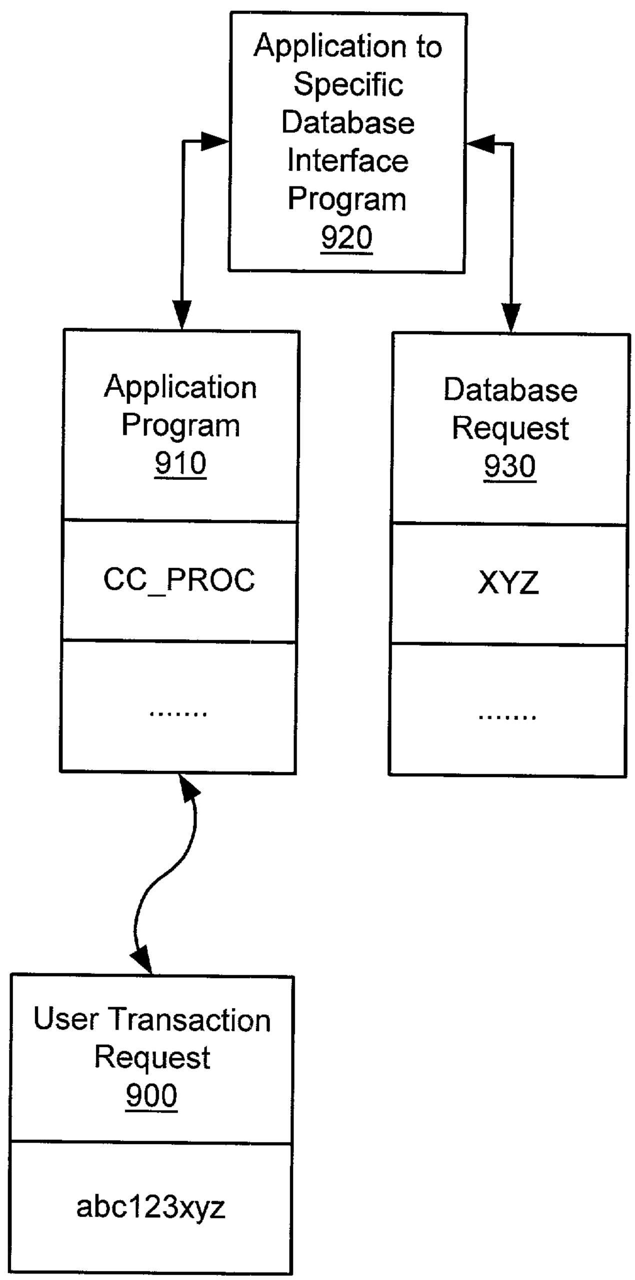

User Defined Key 850
Database Indentifier 860
01DISY
NN...456
....
....
Dynamically Built Access Key For Packageset Table 840
Packageset Input Processing 820
Packageset Output Processing 830
Data From Database 880
Database Access Pointer 870
Application Program 810
CC_PROC
....
FSO Database 890
XYZ
....
User Transaction Request 800
CC_REQ

Figure 8



PRIOR ART

Figure 9

122 124

120

DICTIONARY ELEMENTS	USEABLE IN KEYS?
A	NO
B	NO
C	NO
D	NO
E	NO
F	NO
G	NO
H	NO
I	NO
J	NO
K	NO
L	NO
M	NO
N	YES
O	YES
P	YES
Q	YES
R	YES
S	YES
T	YES
U	YES
V	YES
W	YES
X	YES
Y	YES
Z	YES

FIGURE 10

132 134

130

AVAILABLE KEY ELEMENTS	USEABLE IN KEY?
N	NO
O	NO
P	NO
Q	NO
R	NO
S	NO
T	NO
U	NO
V	YES
W	YES
X	YES
Y	YES
Z	YES

FIGURE 11

142 144 146

140

AVAILABLE KEY ELEMENTS	USE AS KEY FIELD?	SEQUENCE
V	NO	N/A
W	YES	2
X	YES	1
Y	NO	N/A
Z	YES	3

FIGURE 12

152 154 156 158 160

150

KEY FIELD	FIELD SEQUENCE	ELEMENT NAME	FIELD LENGTH	FIELD TYPE
X	1	COMPANY ID	2	NUMERIC
W	2	CREDIT CARD TYPE	3	CHARACTER
Z	3	ON US/NOT ON US	1	CHARACTER

FIGURE 13

172 174 178

ROW	KEY VALUES			PACKAGESET SWITCHING TABLE VALUES
	X	W	Z	DATABASE IDENTIFIER VALUE
1	12	VIS	Y	NN1234567890123456
2	12	DIS	Y	NN1234567890123457
3	12	*	*	NN1234567890123458
4	12	*	Y	NN1234567890123459
5	14	VIS	N	NN1234567890123460
6	14	DIS	N	NN1234567890123461
7	*	*	*	NN1234567890123462

170

175 176 177 179

FIGURE 14

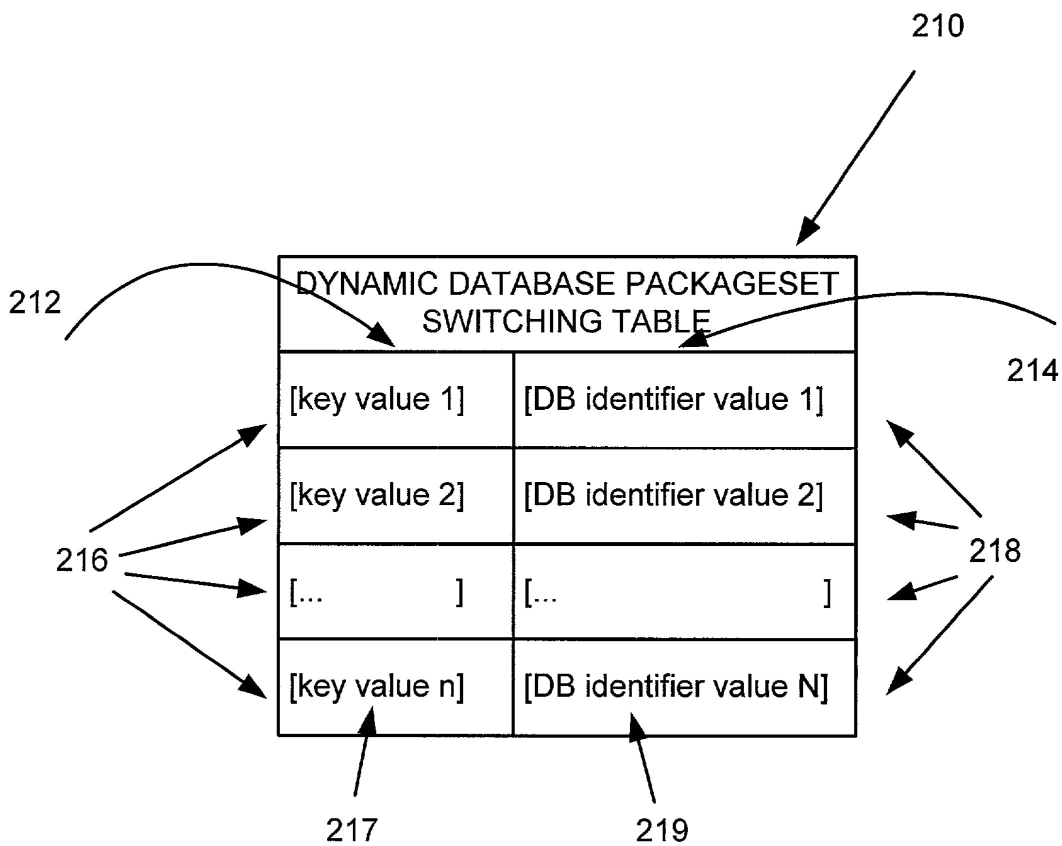

210
212
214
DYNAMIC DATABASE PACKAGESET SWITCHING TABLE
[key value 1]
[DB identifier value 1]
[key value 2]
[DB identifier value 2]
[...   ]
[...   ]
[key value n]
[DB identifier value N]
216
218
217
219

FIGURE 15